

Fundamentals of Machine Learning for Predictive Data Analytics

Chapter 7: Error-based Learning Sections 7.4, 7.5

John Kelleher and Brian Mac Namee and Aoife D'Arcy

john.d.kelleher@dit.ie

brian.macnamee@ucd.ie

aoife@theanalyticsstore.com

- 1 Interpreting Multivariable Linear Regression Models
- 2 Setting the Learning Rate Using Weight Decay
- 3 Handling Categorical Descriptive Features
- 4 Handling Categorical Target Features: Logistic Regression
- 5 Modeling Non-linear Relationships
- 6 Multinomial Logistic Regression
- 7 Support Vector Machines

Interpreting Multivariable Linear Regression Models

- The weights used by linear regression models indicate the effect of each descriptive feature on the predictions returned by the model.
- Both the **sign** and the **magnitude** of the weight provide information on how the descriptive feature effects the predictions of the model.

Table: Weights and standard errors for each feature in the office rentals model.

| Descriptive Feature | Weight | Standard Error | t-statistic | p-value |
|---------------------|----------|----------------|-------------|---------|
| SIZE | 0.6270 | 0.0545 | 11.504 | <0.0001 |
| FLOOR | -0.1781 | 2.7042 | -0.066 | 0.949 |
| BROADBAND RATE | 0.071396 | 0.2969 | 0.240 | 0.816 |

- It is tempting to infer the relative importance of the different descriptive features in the model from the magnitude of the weights
- However, direct comparison of the weights tells us little about their relative importance.
- A better way to determine the importance of each descriptive feature in the model is to perform a **statistical significance test**.

- The statistical significance test we use to analyze the importance of a descriptive feature $\mathbf{d}[j]$ in a linear regression model is the ***t-test***.
- The null hypothesis for this test is that the feature does not have a significant impact on the model. The test statistic we calculate is called the *t*-statistic.

- The standard error for the overall model is calculated as

$$se = \sqrt{\frac{\sum_{i=1}^n (t_i - M_{\mathbf{w}}(\mathbf{d}_i))^2}{n - 2}} \quad (1)$$

- A standard error calculation is then done for a descriptive feature as follows:

$$se(\mathbf{d}[j]) = \frac{se}{\sqrt{\sum_{i=1}^n (\mathbf{d}_i[j] - \bar{\mathbf{d}}[j])^2}} \quad (2)$$

- The t -statistic for this test is calculated as follows:

$$t = \frac{\mathbf{w}[j]}{se(\mathbf{d}[j])} \quad (3)$$

- Using a standard t -statistic look-up table, we can then determine the p -value associated with this test (this is a two tailed t -test with degrees of freedom set to the number of instances in the training set minus 2).
- If the p -value is less than the required significance level, typically 0.05, we reject the null hypothesis and say that the descriptive feature has a significant impact on the model; otherwise we say that it does not.

Table: Weights and standard errors for each feature in the office rentals model.

| Descriptive Feature | Weight | Standard Error | t -statistic | p -value |
|---------------------|----------|----------------|----------------|------------|
| SIZE | 0.6270 | 0.0545 | 11.504 | <0.0001 |
| FLOOR | -0.1781 | 2.7042 | -0.066 | 0.949 |
| BROADBAND RATE | 0.071396 | 0.2969 | 0.240 | 0.816 |

Setting the Learning Rate Using Weight Decay

- **Learning rate decay** allows the learning rate to start at a large value and then decay over time according to a predefined schedule.
- A good approach is to use the following decay schedule:

$$\alpha_\tau = \alpha_0 \frac{c}{c + \tau} \quad (4)$$

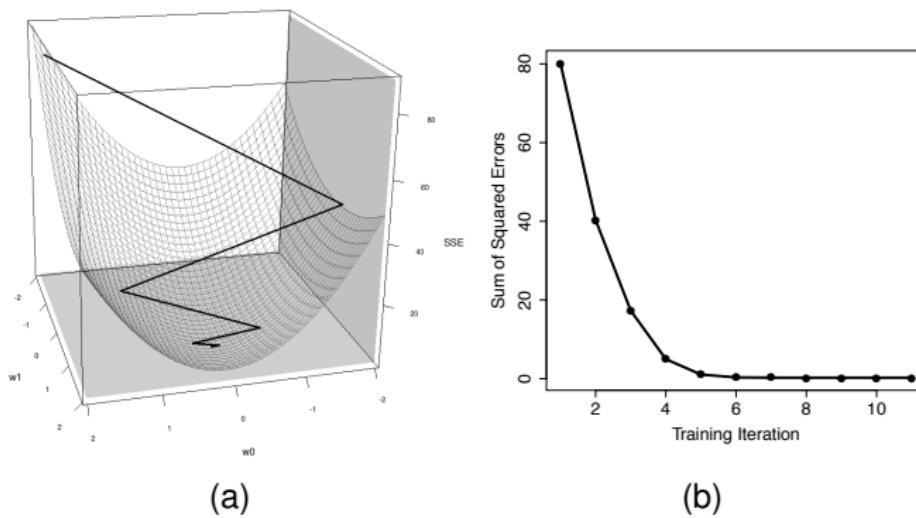


Figure: (a) The journey across the error surface for the office rentals prediction problem when learning rate decay is used ($\alpha_0 = 0.18$, $c = 10$); (b) a plot of the changing sum of squared error values during this journey.

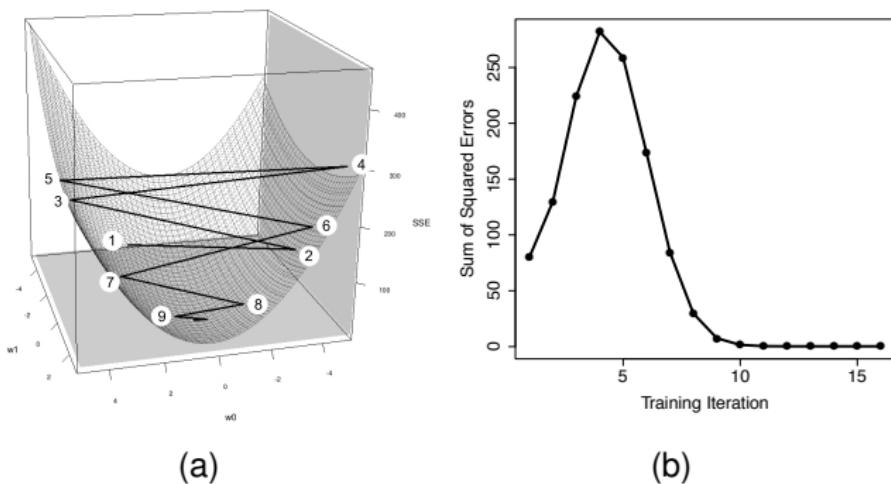


Figure: (a) The journey across the error surface for the office rentals prediction problem when learning rate decay is used ($\alpha_0 = 0.25$, $c = 100$); (b) a plot of the changing sum of squared error values during this journey.

Handling Categorical Descriptive Features

- The basic structure of the multivariable linear regression model allows for only continuous descriptive features, so we need a way to handle categorical descriptive features.
- The most common approach to handling categorical features uses a transformation that converts a single categorical descriptive feature into a number of continuous descriptive feature values that can encode the levels of the categorical feature.
- For example, the ENERGY RATING descriptive feature would be converted into three new continuous descriptive features, as it has 3 distinct levels: 'A', 'B', or 'C'.

Table: The office rentals dataset adjusted to handle the categorical ENERGY RATING descriptive feature in linear regression models.

| ID | SIZE | FLOOR | BROADBAND RATE | ENERGY RATING A | ENERGY RATING B | ENERGY RATING C | RENTAL PRICE |
|----|-------|-------|----------------|-----------------|-----------------|-----------------|--------------|
| 1 | 500 | 4 | 8 | 0 | 0 | 1 | 320 |
| 2 | 550 | 7 | 50 | 1 | 0 | 0 | 380 |
| 3 | 620 | 9 | 7 | 1 | 0 | 0 | 400 |
| 4 | 630 | 5 | 24 | 0 | 1 | 0 | 390 |
| 5 | 665 | 8 | 100 | 0 | 0 | 1 | 385 |
| 6 | 700 | 4 | 8 | 0 | 1 | 0 | 410 |
| 7 | 770 | 10 | 7 | 0 | 1 | 0 | 480 |
| 8 | 880 | 12 | 50 | 1 | 0 | 0 | 600 |
| 9 | 920 | 14 | 8 | 0 | 0 | 1 | 570 |
| 10 | 1 000 | 9 | 24 | 0 | 1 | 0 | 620 |

- Returning to our example, the regression equation for this RENTAL PRICE model would change to

$$\begin{aligned}\text{RENTAL PRICE} = & \mathbf{w}[0] + \mathbf{w}[1] \times \text{SIZE} + \mathbf{w}[2] \times \text{FLOOR} \\ & + \mathbf{w}[3] \times \text{BROADBAND RATE} \\ & + \mathbf{w}[4] \times \text{ENERGY RATING A} \\ & + \mathbf{w}[5] \times \text{ENERGY RATING B} \\ & + \mathbf{w}[6] \times \text{ENERGY RATING C}\end{aligned}$$

where the newly added categorical features allow the original ENERGY RATING feature to be included.

Handling Categorical Target Features: Logistic Regression

Table: A dataset listing features for a number of generators.

| ID | RPM | VIBRATION | STATUS | ID | RPM | VIBRATION | STATUS |
|----|-----|-----------|--------|----|-----|-----------|--------|
| 1 | 568 | 585 | good | 29 | 562 | 309 | faulty |
| 2 | 586 | 565 | good | 30 | 578 | 346 | faulty |
| 3 | 609 | 536 | good | 31 | 593 | 357 | faulty |
| 4 | 616 | 492 | good | 32 | 626 | 341 | faulty |
| 5 | 632 | 465 | good | 33 | 635 | 252 | faulty |
| 6 | 652 | 528 | good | 34 | 658 | 235 | faulty |
| 7 | 655 | 496 | good | 35 | 663 | 299 | faulty |
| 8 | 660 | 471 | good | 36 | 677 | 223 | faulty |
| 9 | 688 | 408 | good | 37 | 685 | 303 | faulty |
| 10 | 696 | 399 | good | 38 | 698 | 197 | faulty |
| 11 | 708 | 387 | good | 39 | 699 | 311 | faulty |
| 12 | 701 | 434 | good | 40 | 712 | 257 | faulty |
| 13 | 715 | 506 | good | 41 | 722 | 193 | faulty |
| 14 | 732 | 485 | good | 42 | 735 | 259 | faulty |
| 15 | 731 | 395 | good | 43 | 738 | 314 | faulty |
| 16 | 749 | 398 | good | 44 | 753 | 113 | faulty |
| 17 | 759 | 512 | good | 45 | 767 | 286 | faulty |
| 18 | 773 | 431 | good | 46 | 771 | 264 | faulty |
| 19 | 782 | 456 | good | 47 | 780 | 137 | faulty |
| 20 | 797 | 476 | good | 48 | 784 | 131 | faulty |
| 21 | 794 | 421 | good | 49 | 798 | 132 | faulty |
| 22 | 824 | 452 | good | 50 | 820 | 152 | faulty |
| 23 | 835 | 441 | good | 51 | 834 | 157 | faulty |
| 24 | 862 | 372 | good | 52 | 858 | 163 | faulty |
| 25 | 879 | 340 | good | 53 | 888 | 91 | faulty |
| 26 | 892 | 370 | good | 54 | 891 | 156 | faulty |
| 27 | 913 | 373 | good | 55 | 911 | 79 | faulty |
| 28 | 933 | 330 | good | 56 | 939 | 99 | faulty |

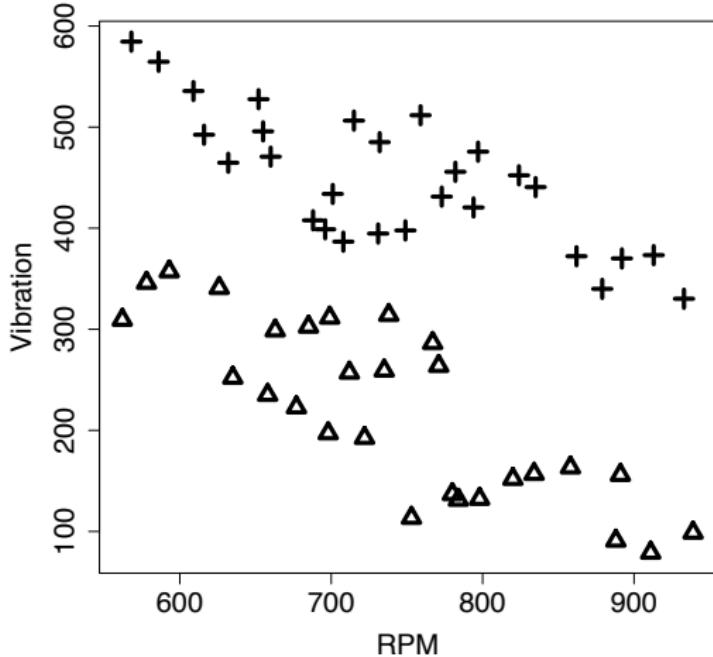


Figure: A scatter plot of the RPM and VIBRATION descriptive features from the generators dataset shown in Table 4 [18] where 'good' generators are shown as crosses and 'faulty' generators are shown as triangles.

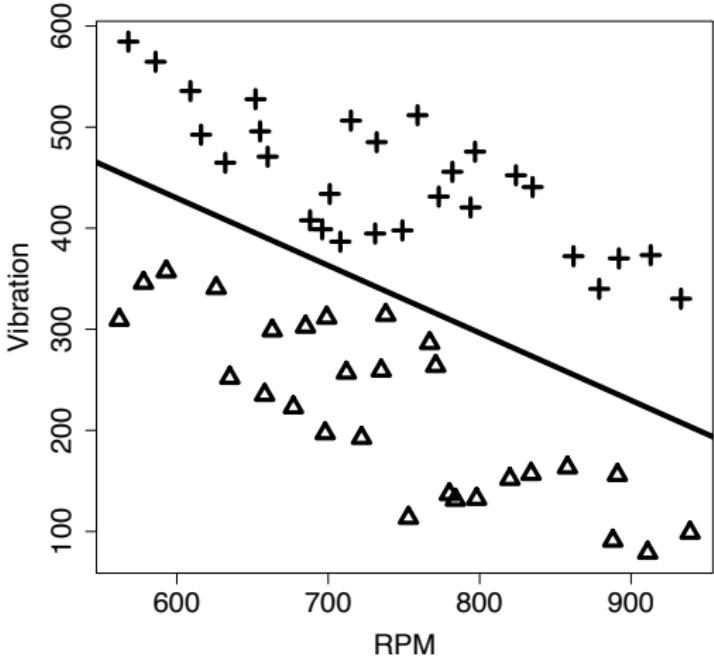


Figure: A scatter plot of the RPM and VIBRATION descriptive features from the generators dataset shown in Table 4 [18]. A decision boundary separating '*good*' generators (crosses) from '*faulty*' generators (triangles) is also shown.

- As the decision boundary is a **linear separator** it can be defined using the equation of the line as:

$$\text{VIBRATION} = 830 - 0.667 \times \text{RPM} \quad (5)$$

or

$$830 - 0.667 \times \text{RPM} - \text{VIBRATION} = 0 \quad (6)$$

- Applying Equation (6)^[21] to the instance $\text{RPM} = 810$, $\text{VIBRATION} = 495$, which is above the decision boundary, gives the following result:

$$830 - 0.667 \times 810 - 495 = -205.27$$

- By contrast, if we apply Equation (6)^[21] to the instance $\text{RPM} = 650$ and $\text{VIBRATION} = 240$, which is below the decision boundary, we get

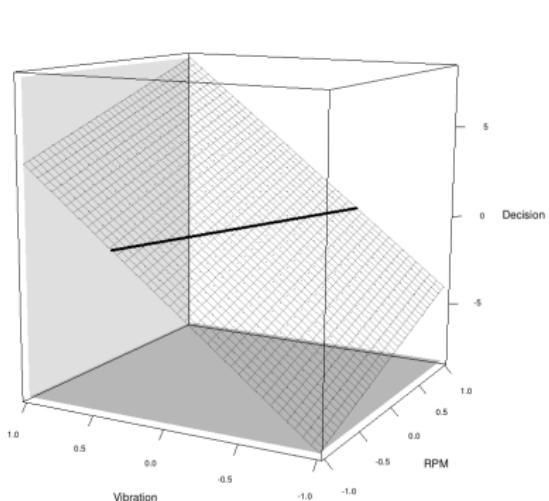
$$830 - 0.667 \times 650 - 240 = 156.45$$

- All the data points above the decision boundary will result in a negative value when plugged into the decision boundary equation, while all data points below the decision boundary will result in a positive value.

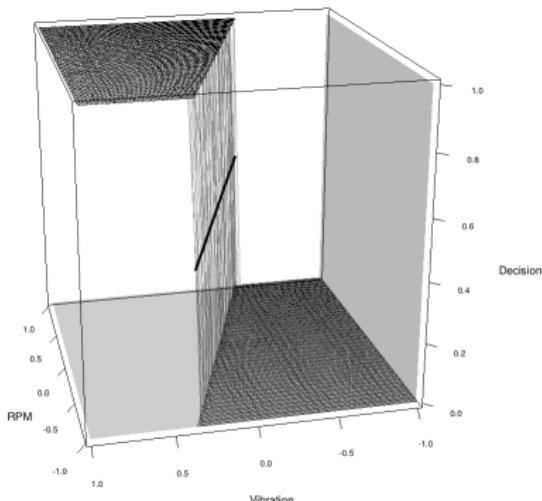
- Reverting to our previous notation we have:

$$\mathbb{M}_{\mathbf{w}}(\mathbf{d}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{d} \geq 0 \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

- The surface defined by this rule is known as a **decision surface**.



(a)



(b)

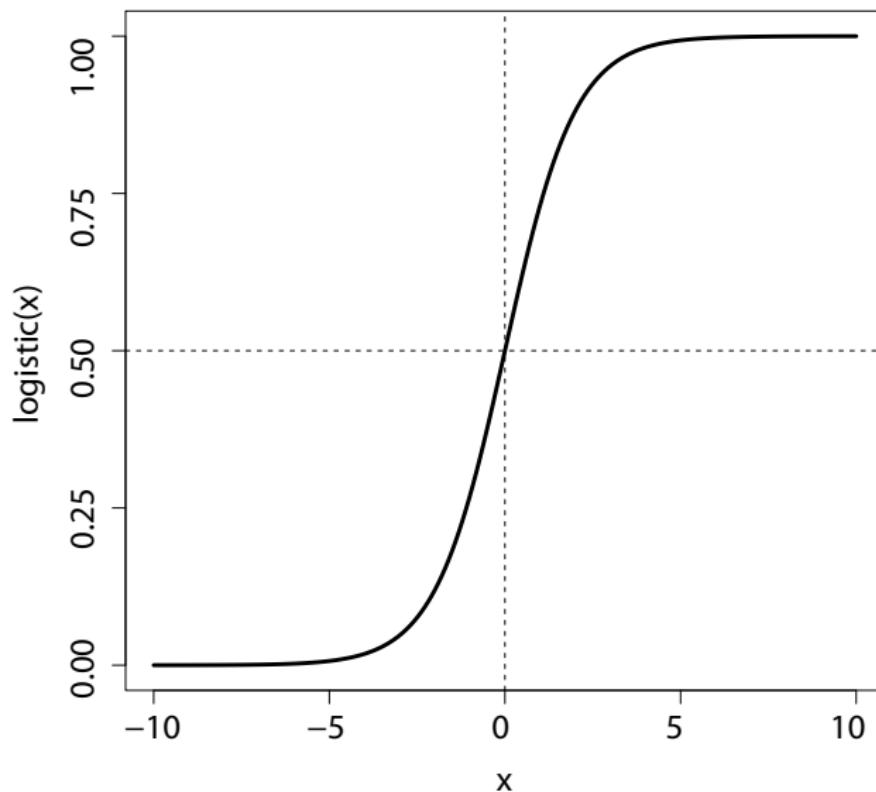
Figure: (a) A surface showing the value of Equation (6)^[21] for all values of RPM and VIBRATION. The decision boundary given in Equation (6)^[21] is highlighted. (b) The same surface linearly thresholded at zero to operate as a predictor.

- The hard decision boundary given in Equation (7)^[24] is **discontinuous** so is not differentiable and so we can't calculate the gradient of the error surface.
- Furthermore, the model always makes completely confident predictions of 0 or 1, whereas a little more subtlety is desirable.
- We address these issues by using a more sophisticated threshold function that is continuous, and therefore differentiable, and that allows for the subtlety desired: the **logistic function**

logistic function

$$\text{Logistic}(x) = \frac{1}{1 + e^{-x}} \quad (8)$$

where x is a numeric value and e is **Euler's number** and is approximately equal to 2.7183.



- To build a logistic regression model, we simply pass the output of the basic linear regression model through the logistic function

$$\begin{aligned} M_{\mathbf{w}}(\mathbf{d}) &= \text{Logistic}(\mathbf{w} \cdot \mathbf{d}) \\ &= \frac{1}{1 + e^{-\mathbf{w} \cdot \mathbf{d}}} \end{aligned} \tag{9}$$

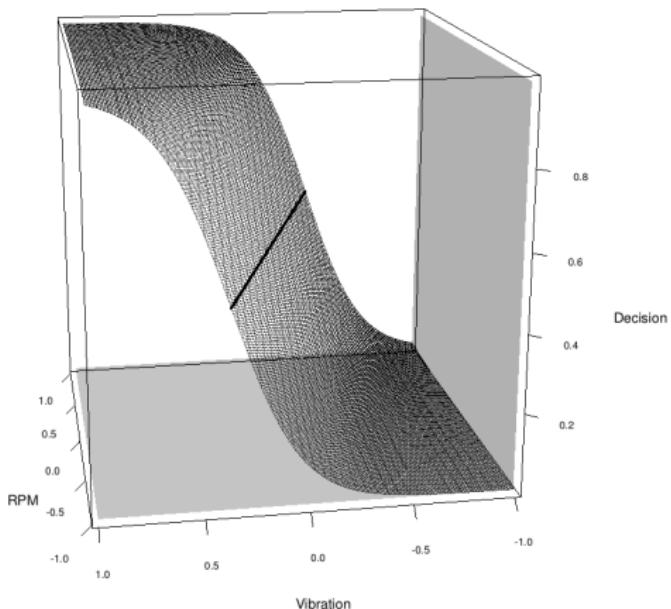
A note on training logistic regression models:

- Before we train a logistic regression model we map the binary target feature levels to 0 or 1.
- The error of the model on each instance is then the difference between the target feature (0 or 1) and the value of the prediction [0, 1].

Example

$$\mathbb{M}_w(\langle \text{RPM}, \text{VIBRATION} \rangle)$$

$$= \frac{1}{1 + e^{(-0.4077 + 4.1697 \times \text{RPM} + 6.0460 \times \text{VIBRATION})}}$$



- The decision surface for the example logistic regression model.

$$P(t = \text{'faulty'} | \mathbf{d}) = \mathbb{M}_{\mathbf{w}}(\mathbf{d})$$

$$P(t = \text{'good'} | \mathbf{d}) = 1 - \mathbb{M}_{\mathbf{w}}(\mathbf{d})$$

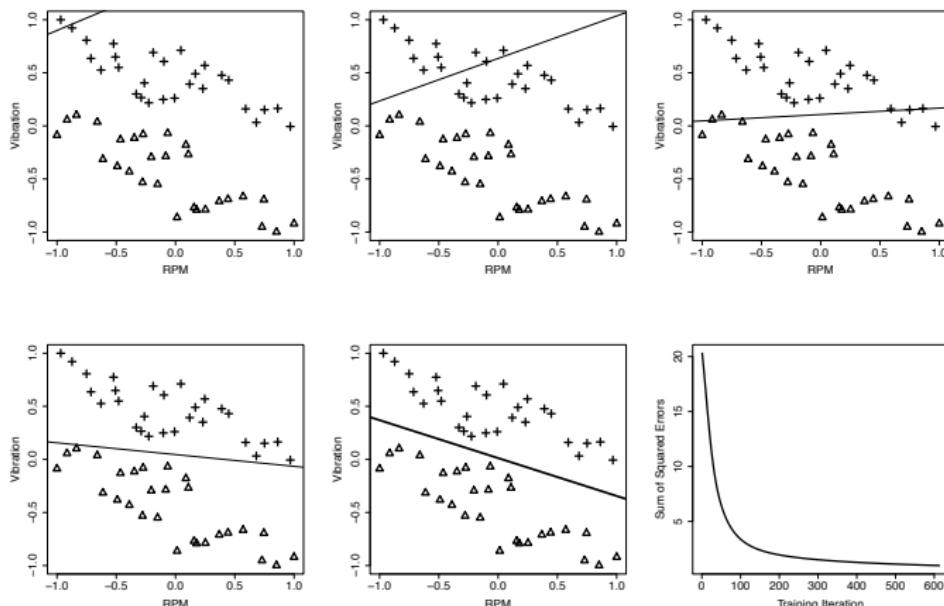


Figure: A selection of the logistic regression models developed during the gradient descent process for the machinery dataset from Table 4 [18]. The bottom-right panel shows the sum of squared error values generated during the gradient descent process.

- To repurpose the gradient descent algorithm for training logistic regression models the only change that needs to be made is i in the weight update rule.
- See pg. 360 in book for details of how to derive the new weight update rule.
- The new weight update rule is:

$$\mathbf{w}[j] \leftarrow \mathbf{w}[j] + \alpha \times \sum_{i=1}^n ((t - M_{\mathbf{w}}(\mathbf{d}_i)) \times M_{\mathbf{w}}(\mathbf{d}_i) \times (1 - M_{\mathbf{w}}(\mathbf{d}_i)) \times \mathbf{d}_i[j])$$

| ID | RPM | VIBRATION | STATUS | ID | RPM | VIBRATION | STATUS |
|----|-----|-----------|--------|----|------|-----------|--------|
| 1 | 498 | 604 | faulty | 35 | 501 | 463 | good |
| 2 | 517 | 594 | faulty | 36 | 526 | 443 | good |
| 3 | 541 | 574 | faulty | 37 | 536 | 412 | good |
| 4 | 555 | 587 | faulty | 38 | 564 | 394 | good |
| 5 | 572 | 537 | faulty | 39 | 584 | 398 | good |
| 6 | 600 | 553 | faulty | 40 | 602 | 398 | good |
| 7 | 621 | 482 | faulty | 41 | 610 | 428 | good |
| 8 | 632 | 539 | faulty | 42 | 638 | 389 | good |
| 9 | 656 | 476 | faulty | 43 | 652 | 394 | good |
| 10 | 653 | 554 | faulty | 44 | 659 | 336 | good |
| 11 | 679 | 516 | faulty | 45 | 662 | 364 | good |
| 12 | 688 | 524 | faulty | 46 | 672 | 308 | good |
| 13 | 684 | 450 | faulty | 47 | 691 | 248 | good |
| 14 | 699 | 512 | faulty | 48 | 694 | 401 | good |
| 15 | 703 | 505 | faulty | 49 | 718 | 313 | good |
| 16 | 717 | 377 | faulty | 50 | 720 | 410 | good |
| 17 | 740 | 377 | faulty | 51 | 723 | 389 | good |
| 18 | 749 | 501 | faulty | 52 | 744 | 227 | good |
| 19 | 756 | 492 | faulty | 53 | 741 | 397 | good |
| 20 | 752 | 381 | faulty | 54 | 770 | 200 | good |
| 21 | 762 | 508 | faulty | 55 | 764 | 370 | good |
| 22 | 781 | 474 | faulty | 56 | 790 | 248 | good |
| 23 | 781 | 480 | faulty | 57 | 786 | 344 | good |
| 24 | 804 | 460 | faulty | 58 | 792 | 290 | good |
| 25 | 828 | 346 | faulty | 59 | 818 | 268 | good |
| 26 | 830 | 366 | faulty | 60 | 845 | 232 | good |
| 27 | 864 | 344 | faulty | 61 | 867 | 195 | good |
| 28 | 882 | 403 | faulty | 62 | 878 | 168 | good |
| 29 | 891 | 338 | faulty | 63 | 895 | 218 | good |
| 30 | 921 | 362 | faulty | 64 | 916 | 221 | good |
| 31 | 941 | 301 | faulty | 65 | 950 | 156 | good |
| 32 | 965 | 336 | faulty | 66 | 956 | 174 | good |
| 33 | 976 | 297 | faulty | 67 | 973 | 134 | good |
| 34 | 994 | 287 | faulty | 68 | 1002 | 121 | good |

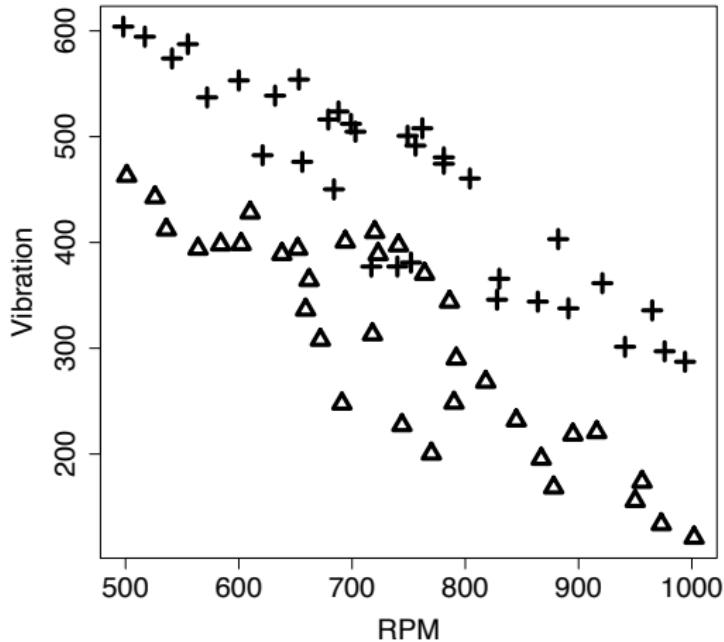


Figure: A scatter plot of the extended generators dataset given in Table 35^[35], which results in instances with the different target levels overlapping with each other. 'good' generators are shown as crosses, and 'faulty' generators are shown as triangles.

- For logistic regression models we recommend that descriptive feature values always be normalized.
- In this example, before the training process begins, both descriptive features are normalized to the range $[-1, 1]$.

- For this example let's assume that:
 - $\alpha = 0.02$

Initial Weights

$w[0]: -2.9465 \quad w[1]: -1.0147 \quad w[2]: -2.1610$

Iteration 1

| ID | TARGET LEVEL | Squared Error | | | $\text{errorDelta}(\mathcal{D}, \mathbf{w}[i])$ | | |
|--------------------------------------|-----------------|---------------|---------|---------|-------------------------------------------------|-----------------|-----------------|
| | | Pred. | Error | Error | $\mathbf{w}[0]$ | $\mathbf{w}[1]$ | $\mathbf{w}[2]$ |
| 1 | 1 | 0.5570 | 0.4430 | 0.1963 | 0.1093 | -0.1093 | 0.1093 |
| 2 | 1 | 0.5168 | 0.4832 | 0.2335 | 0.1207 | -0.1116 | 0.1159 |
| 3 | 1 | 0.4469 | 0.5531 | 0.3059 | 0.1367 | -0.1134 | 0.1197 |
| 4 | 1 | 0.4629 | 0.5371 | 0.2885 | 0.1335 | -0.1033 | 0.1244 |
| ... | | | | | | | |
| 65 | 0 | 0.0037 | -0.0037 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 66 | 0 | 0.0042 | -0.0042 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 67 | 0 | 0.0028 | -0.0028 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 68 | 0 | 0.0022 | -0.0022 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| Sum | | | | 24.4738 | 2.7031 | -0.7015 | 1.6493 |
| Sum of squared errors (Sum/2) | | | | 12.2369 | | | |

$$\mathbf{w}[j] \leftarrow \mathbf{w}[j] + \alpha \times \sum_{i=1}^n ((t_i - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i)) \times \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i) \times (1 - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i)) \times \mathbf{d}_i[j])$$

New Weights (after Iteration 1)

| | | | | | |
|------------------|---------|------------------|---------|------------------|---------|
| $\mathbf{w}[0]:$ | -2.8924 | $\mathbf{w}[1]:$ | -1.0287 | $\mathbf{w}[2]:$ | -2.1940 |
|------------------|---------|------------------|---------|------------------|---------|

Iteration 2

| ID | TARGET LEVEL | Squared Error | | | $\text{errorDelta}(\mathcal{D}, \mathbf{w}[i])$ | | |
|--------------------------------------|-----------------|---------------|---------|---------|-------------------------------------------------|-----------------|-----------------|
| | | Pred. | Error | Error | $\mathbf{w}[0]$ | $\mathbf{w}[1]$ | $\mathbf{w}[2]$ |
| 1 | 1 | 0.5817 | 0.4183 | 0.1749 | 0.1018 | -0.1018 | 0.1018 |
| 2 | 1 | 0.5414 | 0.4586 | 0.2103 | 0.1139 | -0.1053 | 0.1094 |
| 3 | 1 | 0.4704 | 0.5296 | 0.2805 | 0.1319 | -0.1094 | 0.1155 |
| 4 | 1 | 0.4867 | 0.5133 | 0.2635 | 0.1282 | -0.0992 | 0.1194 |
| ... | | | | | | | |
| 65 | 0 | 0.0037 | -0.0037 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 66 | 0 | 0.0043 | -0.0043 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 67 | 0 | 0.0028 | -0.0028 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 68 | 0 | 0.0022 | -0.0022 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| | | Sum | | 24.0524 | 2.7236 | -0.6646 | 1.6484 |
| Sum of squared errors (Sum/2) | | | | 12.0262 | | | |

$$\mathbf{w}[j] \leftarrow \mathbf{w}[j] + \alpha \times \sum_{i=1}^n ((t_i - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i)) \times \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i) \times (1 - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i)) \times \mathbf{d}_i[j])$$

New Weights (after Iteration 2)

| | | | | | |
|------------------|---------|------------------|---------|------------------|---------|
| $\mathbf{w}[0]:$ | -2.8380 | $\mathbf{w}[1]:$ | -1.0416 | $\mathbf{w}[2]:$ | -2.2271 |
|------------------|---------|------------------|---------|------------------|---------|

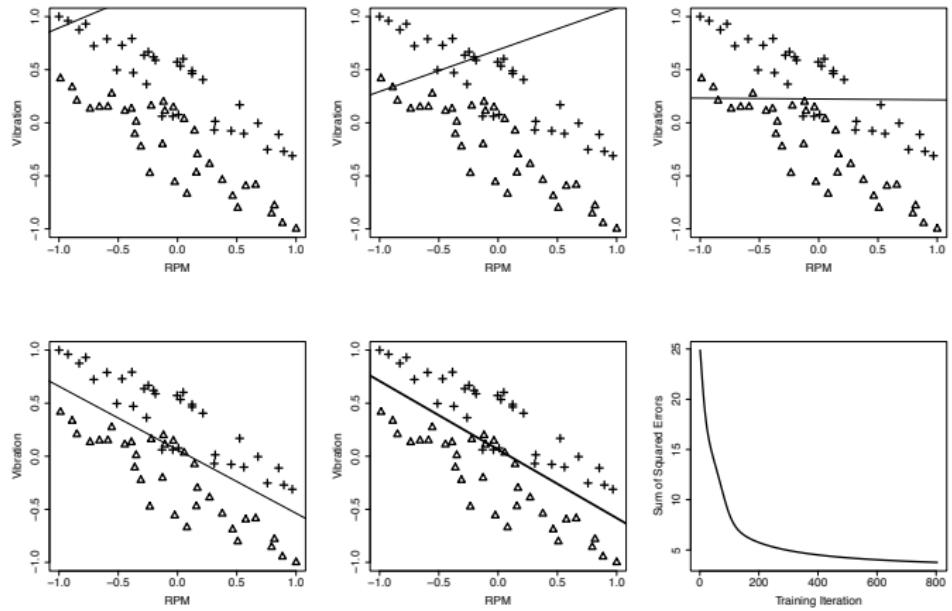


Figure: A selection of the logistic regression models developed during the gradient descent process for the extended generators dataset in Table 35 [35]. The bottom-right panel shows the sum of squared error values generated during the gradient descent process.

- The final model found is:

$$\mathbb{M}_w(\langle \text{RPM}, \text{VIBRATION} \rangle)$$

$$= \frac{1}{1 + e^{(-0.4077 + 4.1697 \times \text{RPM} + 6.0460 \times \text{VIBRATION})}}$$

Modeling Non-linear Relationships

Table: A dataset describing grass growth on Irish farms during July 2012.

| ID | RAIN | GROWTH | ID | RAIN | GROWTH | ID | RAIN | GROWTH |
|----|-------|--------|----|-------|--------|----|-------|--------|
| 1 | 2.153 | 14.016 | 12 | 3.754 | 11.420 | 23 | 3.960 | 10.307 |
| 2 | 3.933 | 10.834 | 13 | 2.809 | 13.847 | 24 | 3.592 | 12.069 |
| 3 | 1.699 | 13.026 | 14 | 1.809 | 13.757 | 25 | 3.451 | 12.335 |
| 4 | 1.164 | 11.019 | 15 | 4.114 | 9.101 | 26 | 1.197 | 10.806 |
| 5 | 4.793 | 4.162 | 16 | 2.834 | 13.923 | 27 | 0.723 | 7.822 |
| 6 | 2.690 | 14.167 | 17 | 3.872 | 10.795 | 28 | 1.958 | 14.010 |
| 7 | 3.982 | 10.190 | 18 | 2.174 | 14.307 | 29 | 2.366 | 14.088 |
| 8 | 3.333 | 13.525 | 19 | 4.353 | 8.059 | 30 | 1.530 | 12.701 |
| 9 | 1.942 | 13.899 | 20 | 3.684 | 12.041 | 31 | 0.847 | 9.012 |
| 10 | 2.876 | 13.949 | 21 | 2.140 | 14.641 | 32 | 3.843 | 10.885 |
| 11 | 4.277 | 8.643 | 22 | 2.783 | 14.138 | 33 | 0.976 | 9.876 |

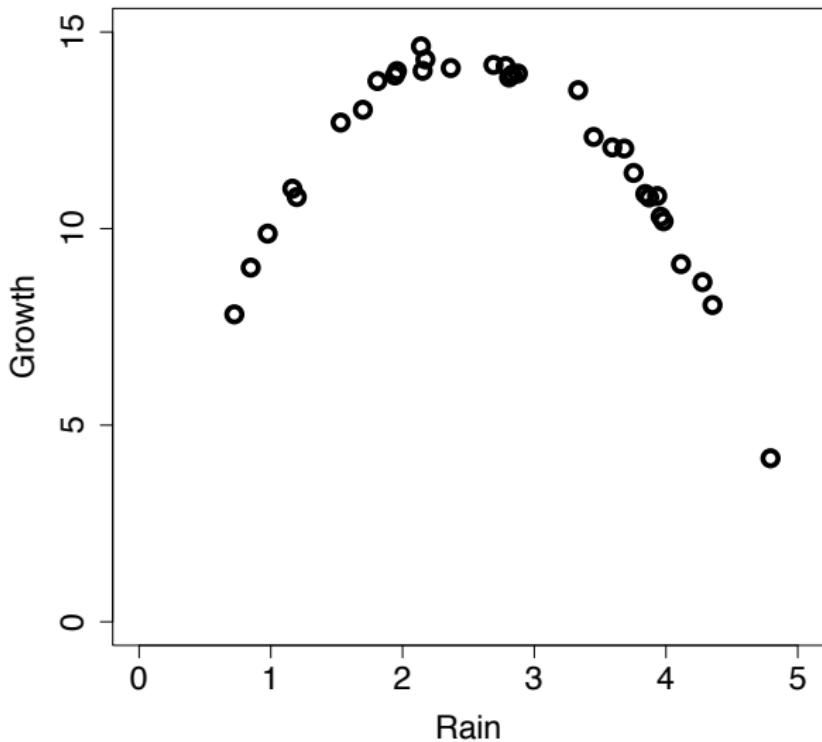


Figure: A scatter plot of the RAIN and GROWTH feature from the grass growth dataset.

- The best linear model we can learn for this data is:

$$\text{GROWTH} = 13.510 + -0.667 \times \text{RAIN}$$

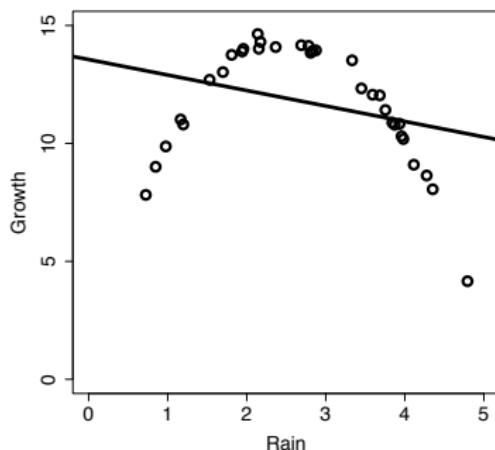


Figure: A simple linear regression model trained to capture the relationship between the grass growth and rainfall.

- In order to handle non-linear relationships we transform the data rather than the model using a set of basis functions:

$$\mathbb{M}_{\mathbf{w}}(\mathbf{d}) = \sum_{k=0}^b \mathbf{w}[k] \times \phi_k(\mathbf{d}) \quad (10)$$

where ϕ_0 to ϕ_b are a series of b basis functions that each transform the input vector \mathbf{d} in a different way.

- The advantage of this is that, except for introducing the mechanism of basis functions, we do not need to make any other changes to the approach we have presented so far.

- The relationship between rainfall and grass growth in the grass growth dataset can be accurately represented as a **second order polynomial** through the following model:

$$\text{GROWTH} = \mathbf{w}[0] \times \phi_0(\text{RAIN}) + \mathbf{w}[1] \times \phi_1(\text{RAIN}) + \mathbf{w}[2] \times \phi_2(\text{RAIN})$$

where

$$\phi_0(\text{RAIN}) = 1$$

$$\phi_1(\text{RAIN}) = \text{RAIN}$$

$$\phi_2(\text{RAIN}) = \text{RAIN}^2$$

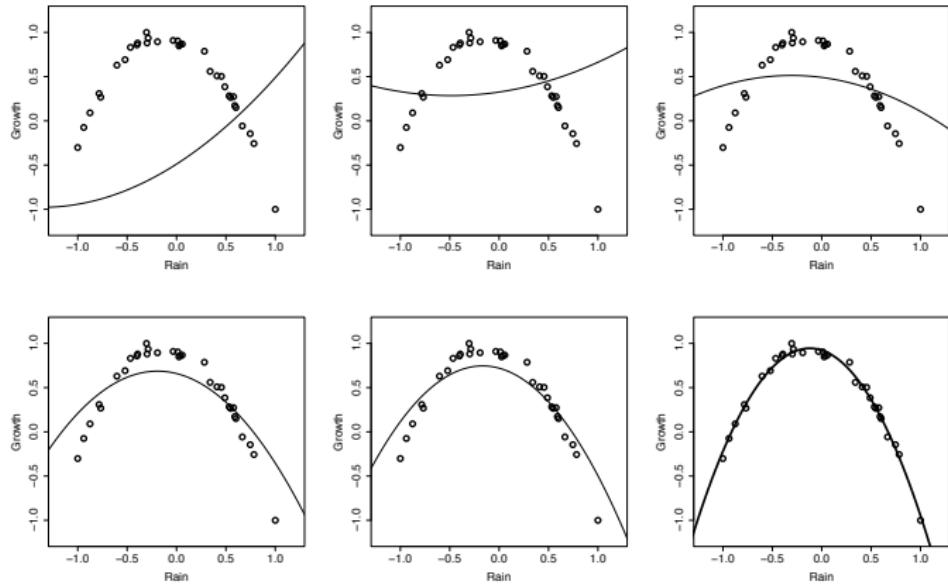
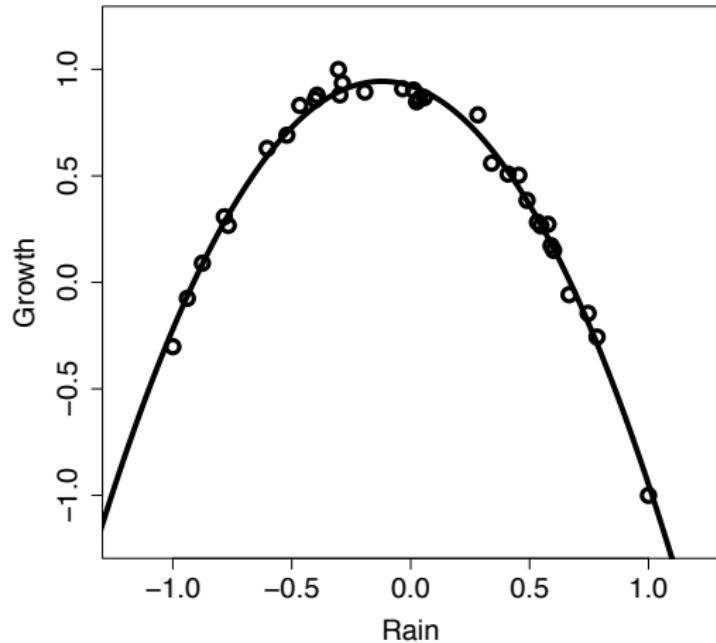


Figure: A selection of the models developed during the gradient descent process for the grass growth dataset from Table 5 [46]. (Note that the RAIN and GROWTH features have been **range normalized** to the $[-1, 1]$ range.)



$$\text{GROWTH} = 0.3707 \times \phi_0(\text{RAIN}) + 0.8475 \times \phi_1(\text{RAIN}) + -1.717 \times \phi_2(\text{RAIN})$$

$$\text{GROWTH} = 0.3707 \times \phi_0(\text{RAIN}) + 0.8475 \times \phi_1(\text{RAIN}) + -1.717 \times \phi_2(\text{RAIN})$$

$$\phi_0(\text{RAIN}) = 1$$

$$\phi_1(\text{RAIN}) = \text{RAIN}$$

$$\phi_2(\text{RAIN}) = \text{RAIN}^2$$

- What is the predicted growth for the following RAIN values:
 - ➊ RAIN= -0.75
 - ➋ RAIN= 0.1
 - ➌ RAIN= 0.9

- Basis functions can also be used for
 - ➊ multivariable simple linear regression models in the same way, the only extra requirement being the definition of more basis functions.
 - ➋ to train logistic regression models for categorical prediction problems that involve non-linear relationships.

Table: A dataset showing participants' responses to viewing '*positive*' and '*negative*' images measured on the EEG P20 and P45 potentials.

| ID | P20 | P45 | TYPE | ID | P20 | P45 | TYPE |
|----|--------|--------|----------|----|--------|--------|----------|
| 1 | 0.4497 | 0.4499 | negative | 26 | 0.0656 | 0.2244 | positive |
| 2 | 0.8964 | 0.9006 | negative | 27 | 0.6336 | 0.2312 | positive |
| 3 | 0.6952 | 0.3760 | negative | 28 | 0.4453 | 0.4052 | positive |
| 4 | 0.1769 | 0.7050 | negative | 29 | 0.9998 | 0.8493 | positive |
| 5 | 0.6904 | 0.4505 | negative | 30 | 0.9027 | 0.6080 | positive |
| 6 | 0.7794 | 0.9190 | negative | 31 | 0.3319 | 0.1473 | positive |
| | ⋮ | | | | | ⋮ | |

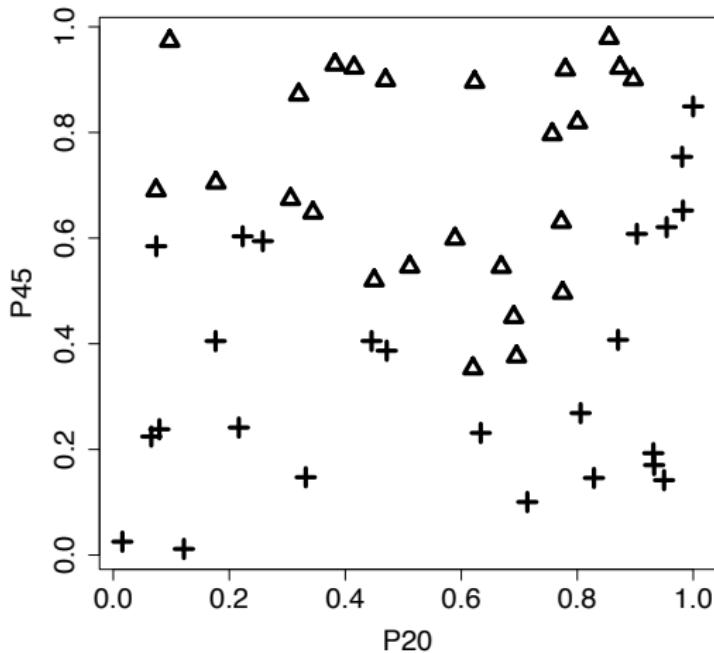


Figure: A scatter plot of the P20 and P45 features from the EEG dataset. 'positive' images are shown as crosses, and 'negative' images are shown as triangles.

- A logistic regression model using basis functions is defined as follows:

$$\mathbb{M}_{\mathbf{w}}(\mathbf{d}) = \frac{1}{1 + e^{-\left(\sum_{j=0}^b \mathbf{w}[j]\phi_j(\mathbf{d})\right)}} \quad (11)$$

- We will use the following basis functions for the EEG problem:

$$\phi_0(\langle P20, P45 \rangle) = 1 \quad \phi_4(\langle P20, P45 \rangle) = P45^2$$

$$\phi_1(\langle P20, P45 \rangle) = P20 \quad \phi_5(\langle P20, P45 \rangle) = P20^3$$

$$\phi_2(\langle P20, P45 \rangle) = P45 \quad \phi_6(\langle P20, P45 \rangle) = P45^3$$

$$\phi_3(\langle P20, P45 \rangle) = P20^2 \quad \phi_7(\langle P20, P45 \rangle) = P20 \times P45$$

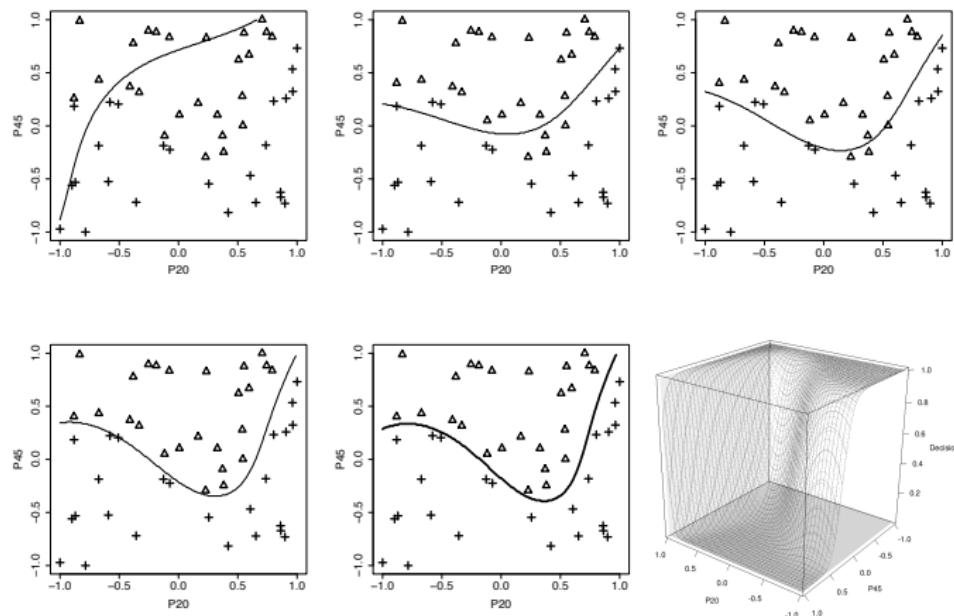
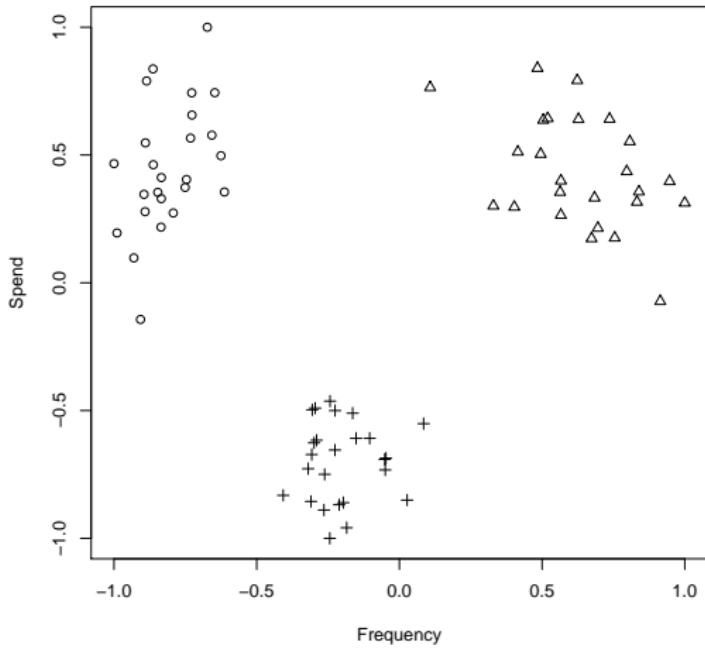
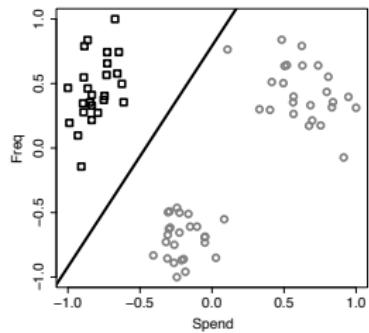


Figure: A selection of the models developed during the gradient descent process for the EEG dataset from Table 6^[55]. The final panel shows the decision surface generated.

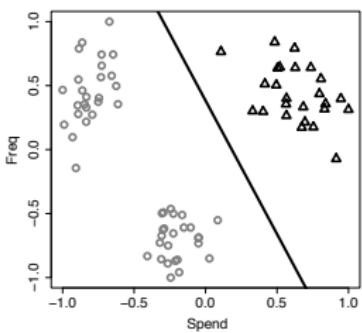
Multinomial Logistic Regression

Table: A dataset of customers of a large national retail chain.

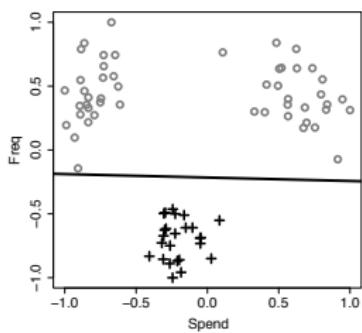




(a)



(b)



(c)

Figure: An illustration of three different **one-versus-all** prediction models for the customer type dataset in Table 7^[61] that has three target levels '*single*' (squares), '*business*' (triangles) and '*family*' (crosses).

- For r target feature levels, we build r separate logistic regression models M_{w_1} to M_{w_r} :

$$M_{w_1}(\mathbf{d}) = \text{logistic}(\mathbf{w}_1 \cdot \mathbf{d})$$

$$M_{w_2}(\mathbf{d}) = \text{logistic}(\mathbf{w}_2 \cdot \mathbf{d})$$

(12)

⋮

$$M_{w_r}(\mathbf{d}) = \text{logistic}(\mathbf{w}_r \cdot \mathbf{d})$$

where M_{w_1} to M_{w_r} are r different one-versus-all logistic regression models, and \mathbf{w}_1 to \mathbf{w}_r are r different sets of weights.

- To combine the outputs of these different models, we normalize their results using:

$$\mathbb{M}'_{\mathbf{w}_k}(\mathbf{d}) = \frac{\mathbb{M}_{\mathbf{w}_k}(\mathbf{d})}{\sum_{l \in \text{levels}(t)} \mathbb{M}_{\mathbf{w}_l}(\mathbf{d})} \quad (13)$$

where $\mathbb{M}'_{\mathbf{w}_k}(\mathbf{d})$ is a revised, normalized prediction for the one-versus-all model for the target level k .

- The r one-versus-all logistic regression models used are trained in **parallel**, and the **revised model outputs**, $\mathbb{M}'_{\mathbf{w}_k}(\mathbf{d})$, are used when calculating the sum of squared errors for each model during the training process.
- This means that the sum of squared errors function is changed slightly to

$$L_2(\mathbb{M}_{\mathbf{w}_k}, \mathcal{D}) = \frac{1}{2} \sum_{i=1}^n (t_i - \mathbb{M}'_{\mathbf{w}_k}(\mathbf{d}_i [1]))^2 \quad (14)$$

- The revised predictions are also used when making predictions for query instances. The predicted level for a query, \mathbf{q} , is the level associated with the one-versus-all model that outputs the highest result after normalization.
- We can write this as

$$\mathbb{M}(\mathbf{q}) = \operatorname{argmax}_{l \in \text{levels}(t)} \mathbb{M}'_{\mathbf{w}_l}(\mathbf{q}) \quad (15)$$

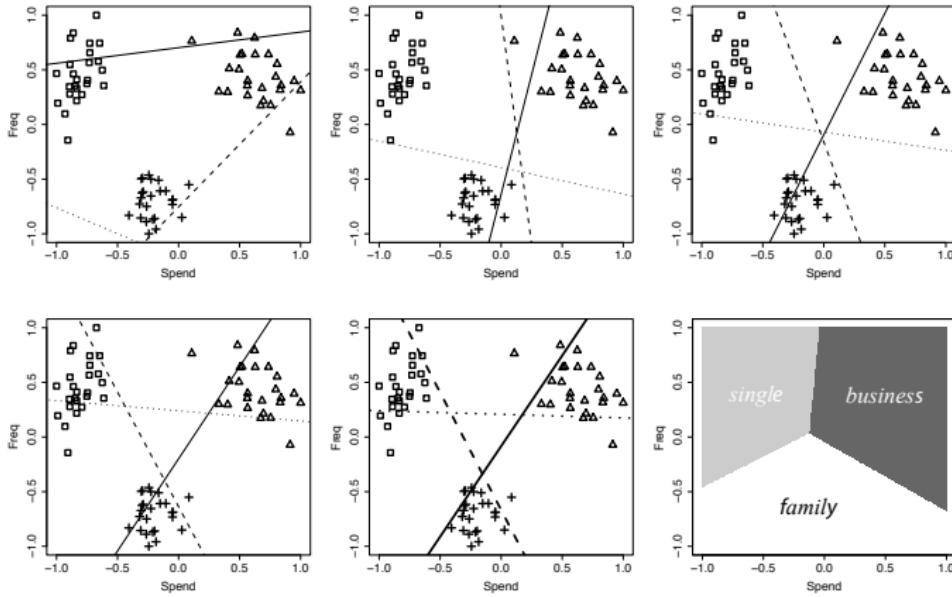
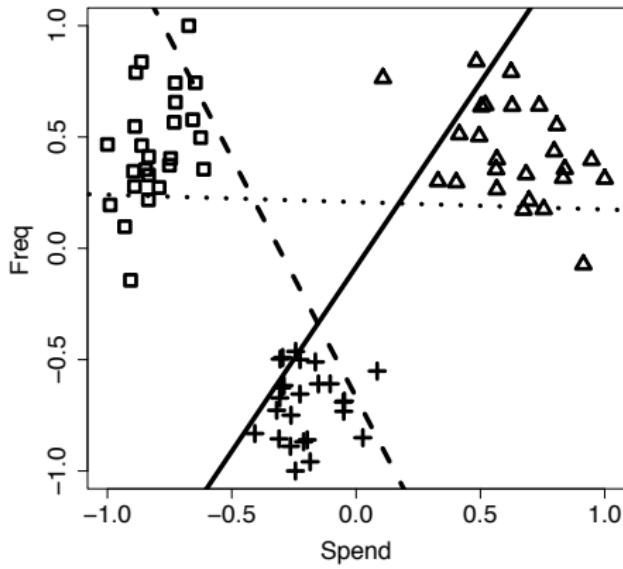


Figure: A selection of the models developed during the gradient descent process for the customer group dataset from Table 7 [61]. Squares represent instances with the '*single*' target level, triangles the '*business*' level and crosses the '*family*' level. (f) illustrates the overall decision boundaries that are learned between the three target levels.



$$\mathbb{M}_{\mathbf{w}'_{single}}(\mathbf{q}) = \text{Logistic}(0.7993 - 15.9030 \times \text{SPEND} + 9.5974 \times \text{FREQ})$$

$$\mathbb{M}_{\mathbf{w}'_{family}}(\mathbf{q}) = \text{Logistic}(3.6526 + -0.5809 \times \text{SPEND} - 17.5886 \times \text{FREQ})$$

$$\mathbb{M}_{\mathbf{w}'_{business}}(\mathbf{q}) = \text{Logistic}(4.6419 + 14.9401 \times \text{SPEND} - 6.9457 \times \text{FREQ})$$

- For a query instance with $\text{SPEND} = 25.67$ and $\text{FREQ} = 6.12$, which are normalized to $\text{SPEND} = -0.7279$ and $\text{FREQ} = 0.4789$, the predictions of the individual models would be

$$\begin{aligned}\mathbb{M}_{\mathbf{w}'_{single}}(\mathbf{q}) &= \text{Logistic}(0.7993 - 15.9030 \times (-0.7279) + 9.5974 \times 0.4789) \\ &= 0.9999\end{aligned}$$

$$\begin{aligned}\mathbb{M}_{\mathbf{w}'_{family}}(\mathbf{q}) &= \text{Logistic}(3.6526 + -0.5809 \times (-0.7279) - 17.5886 \times 0.4789) \\ &= 0.01278\end{aligned}$$

$$\mathbb{M}_{\mathbf{w}'_{business}}(\mathbf{q}) = ?$$

- For a query instance with $\text{SPEND} = 25.67$ and $\text{FREQ} = 6.12$, which are normalized to $\text{SPEND} = -0.7279$ and $\text{FREQ} = 0.4789$, the predictions of the individual models would be

$$\begin{aligned}M_{\mathbf{w}_{single}}(\mathbf{q}) &= \text{Logistic}(0.7993 - 15.9030 \times (-0.7279) + 9.5974 \times 0.4789) \\&= 0.9999\end{aligned}$$

$$\begin{aligned}M_{\mathbf{w}_{family}}(\mathbf{q}) &= \text{Logistic}(3.6526 + -0.5809 \times (-0.7279) - 17.5886 \times 0.4789) \\&= 0.01278\end{aligned}$$

$$\begin{aligned}M_{\mathbf{w}_{business}}(\mathbf{q}) &= \text{Logistic}(4.6419 + 14.9401 \times (-0.7279) - 6.9457 \times 0.4789) \\&= 0.0518\end{aligned}$$

- These predictions would be normalized as follows:

$$\mathbb{M}'_{\mathbf{w}'_{single}}(\mathbf{q}) = \frac{0.9999}{0.9999 + 0.01278 + 0.0518} = 0.9393$$

$$\mathbb{M}'_{\mathbf{w}'_{family}}(\mathbf{q}) = \frac{0.01278}{0.9999 + 0.01278 + 0.0518} = 0.0120$$

$$\mathbb{M}'_{\mathbf{w}'_{business}}(\mathbf{q}) = \frac{0.0518}{0.9999 + 0.01278 + 0.0518} = 0.0487$$

- This means the overall prediction for the query instance is '*single*', as this gets the highest normalized score.

Support Vector Machines

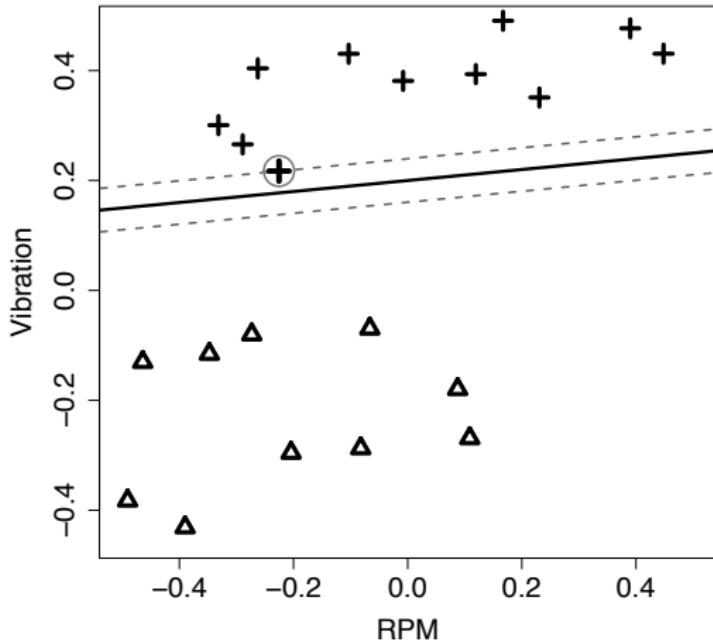


Figure: A small sample of the generators dataset with two features, RPM and VIBRATION, and two target levels, '*good*' (shown as crosses) and '*bad*' (shown as triangles). A decision boundary with a very small margin.

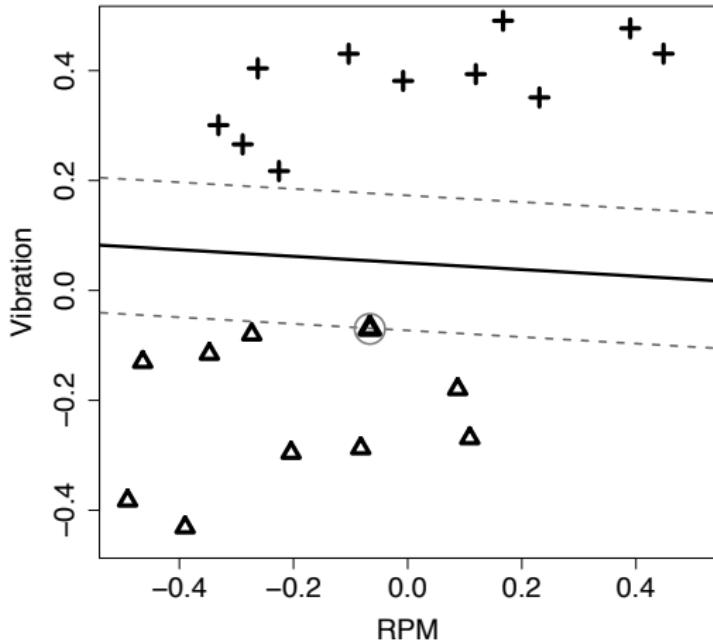


Figure: A small sample of the generators dataset with two features, RPM and VIBRATION, and two target levels, '*good*' (shown as crosses) and '*bad*' (shown as triangles). A decision boundary with a large margin.

- Training a support vector machine involves searching for the decision boundary, or **separating hyperplane**, that leads to the maximum margin as this will best separate the levels of the target feature.
- The instances in a training dataset that fall along the margin extents, and so define the margins, are known as the **support vectors** and define the decision boundary.

- We define the separating hyperplane as follows:

$$w_0 + \mathbf{w} \cdot \mathbf{d} = 0 \quad (16)$$

- For instances above a separating hyperplane

$$w_0 + \mathbf{w} \cdot \mathbf{d} > 0$$

and for instances below a separating hyperplane

$$w_0 + \mathbf{w} \cdot \mathbf{d} < 0$$

- We build a support vector machine prediction model so that instances with the negative target level result in the model outputting ≤ -1 and instances with the positive target level result in the model outputting $\geq +1$.
- The space between the outputs of -1 and $+1$ allows for the margin.

- A support vector machine model is defined as

$$\mathbb{M}_{\alpha, w_0}(\mathbf{q}) = \sum_{i=1}^s (t_i \times \alpha[i] \times (\mathbf{d}_i \cdot \mathbf{q}) + w_0) \quad (17)$$

where

- \mathbf{q} is the set of descriptive features for a query instance;
- $(\mathbf{d}_1, t_1), \dots, (\mathbf{d}_s, t_s)$ are s support vectors (instances composed of descriptive features and a target feature);
- w_0 is the first weight of the decision boundary;
- and α is a set of parameters determined during the training process (there is a parameter for each support vector $\alpha[1], \dots, \alpha[s]$).¹

¹These parameters are formally known as **Lagrange multipliers**.

- Training a support vector machine frames as a **constrained quadratic optimization problem**
- This type of problem is defined in terms of:
 - ➊ a set of constraints
 - ➋ an optimization criterion.

- The required **constraints** required by the training process are

$$w_0 + \mathbf{w} \cdot \mathbf{d} \leq -1 \text{ for } t_i = -1 \quad (18)$$

and:

$$w_0 + \mathbf{w} \cdot \mathbf{d} \geq +1 \text{ for } t_i = +1 \quad (19)$$

- We can combine these two constraints into a single constraint (remember t_i is always equal to either -1 or $+1$):

$$t_i \times (w_0 + \mathbf{w} \cdot \mathbf{d}) \geq 1 \quad (20)$$

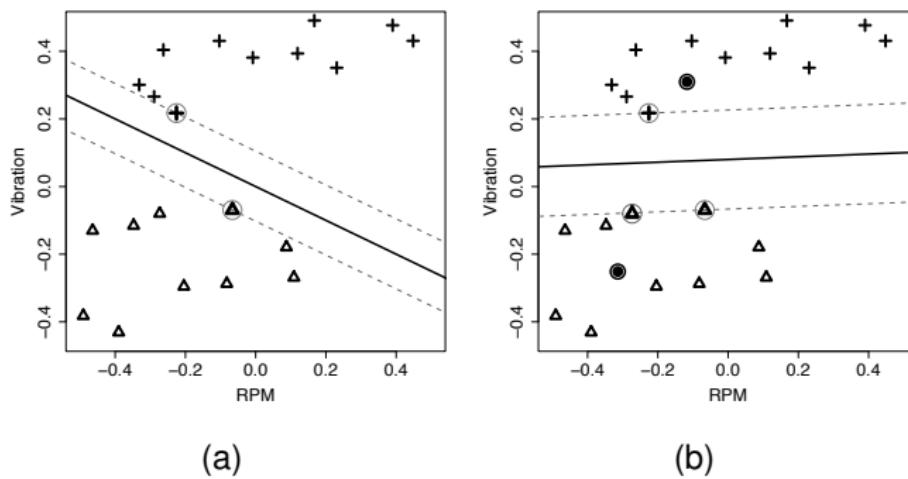


Figure: Different margins that satisfy the constraint in Equation (20)^[81]. The instances that define the margin are highlighted in each case. (b) shows the maximum margin and also shows two query instances represented as black dots.

- The **optimization** criterion used is defined in terms of the perpendicular distance from any instance to the decision boundary and is given by

$$dist(\mathbf{d}) = \frac{w_0 + \text{abs}(\mathbf{w} \cdot \mathbf{d})}{\|\mathbf{w}\|}$$

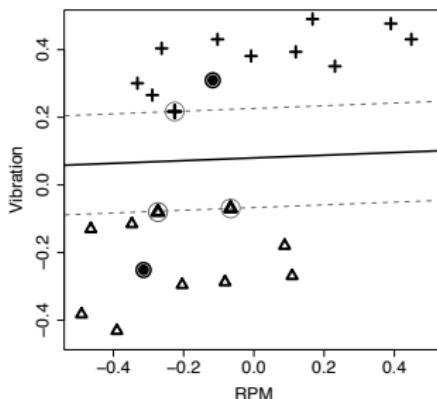
where $\|\mathbf{w}\|$ is known as the **Euclidean norm** of \mathbf{w} and is calculated as

$$\|\mathbf{w}\| = \sqrt{\mathbf{w}[1]^2 + \mathbf{w}[2]^2 + \dots + \mathbf{w}[m]^2}$$

- For instances along the **margin extents**,
 $\text{abs}(\mathbf{w} \cdot \mathbf{d} + w_0) = 1$.
- So, the distance from any instance along the margin extents to the decision boundary is $\frac{1}{\|\mathbf{w}\|}$, and because the margin is symmetrical to either side of the decision boundary, the size of the margin is $\frac{2}{\|\mathbf{w}\|}$.

- The goal when training a support vector machine is
 - maximize $\frac{2}{\|\mathbf{w}\|}$
 - subject to the constraint

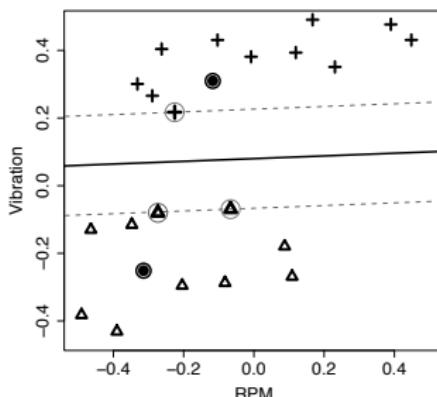
$$t_i \times (w_0 + \mathbf{w} \cdot \mathbf{d}) \geq 1$$



- The optimal decision boundary and associated support vectors for the example we have been following
- In this case '*good*' is the positive level and set to +1, and '*faulty*' is the negative level and set to -1.

- The descriptive feature values and target feature values for the support vectors in these cases are
 - $(\langle -0.225, 0.217 \rangle, +1)$,
 - $(\langle -0.066, -0.069 \rangle, -1)$,
 - $(\langle -0.273, -0.080 \rangle, -1)$.
- The value of w_0 is -0.1838 ,
- The values of the α parameters are

$$\langle 22.056, 6.998, 16.058 \rangle.$$



- The plot shows the position of two new query instances for this problem.
- The descriptive feature values for these queries are
 - $q_1 = \langle -0.314, -0.251 \rangle$
 - $q_2 = \langle -0.117, 0.31 \rangle$.

- For the first query instance, $\mathbf{q}_1 = \langle -0.314, -0.251 \rangle$, the output of the support vector machine model is:

$$\mathbb{M}_{\alpha, w_0}(\mathbf{q}_1)$$

$$\begin{aligned}&= (1 \times 23.056 \times ((-0.225 \times -0.314) + (0.217 \times -0.251))) - 0.1838 \\&\quad + (-1 \times 6.998 \times ((-0.066 \times -0.314) + (-0.069 \times -0.251))) - 0.1838 \\&\quad + (-1 \times 16.058 \times ((-0.273 \times -0.314) + (-0.080 \times -0.251))) - 0.1838 \\&= -2.145\end{aligned}$$

- The model output is less than -1 , so this query is predicted to be a '*faulty*' generator.
- For the second query instance, the model output is 1.592 , so this instance is predicted to be a '*good*' generator.

- **Basis functions** can be used with support vector machines to handle data that is not **linearly separable**
- To use basis functions we update Equation (20)^[81] to

$$t_i \times (w_0 + \mathbf{w} \cdot \phi(\mathbf{d})) \geq 1 \text{ for all } i \quad (21)$$

where ϕ is a set of basis functions applied to the descriptive features \mathbf{d} , and \mathbf{w} is a set of weights containing one weight for each member of ϕ .

- Typically, the number of basis functions in ϕ is larger than the number of descriptive features, so the application of the basis functions moves the data into a higher-dimensional space.
- The expectation is that a linear separating hyperplane will exist in this higher-dimensional space even though it does not in the original feature space.

- The prediction model in this case becomes

$$\mathbb{M}_{\alpha, \phi, w_0}(\mathbf{q}) = \sum_{i=1}^s (t_i \times \alpha[i] \times (\phi(\mathbf{d}_i) \cdot \phi(\mathbf{q})) + w_0) \quad (22)$$

- This equation requires a dot product calculation between the result of applying the basis functions to the query instance and to each of the support vectors which is repeated multiple times during the training process.

- A dot product is a computationally expensive operation,
- We can use a clever trick is used to avoid it:
 - the same result obtained by calculating the dot product of the descriptive features of a support vector and a query instance after having applied the basis functions can be obtained by applying a much less costly **kernel function**, *kernel*, to the original descriptive feature values of the support vector and the query.

- The prediction equation becomes

$$\mathbb{M}_{\alpha, \text{kernel}, w_0}(\mathbf{q}) = \sum_{i=1}^s (t_i \times \alpha[i] \times \text{kernel}(\mathbf{d}_i, \mathbf{q}) + w_0) \quad (23)$$

- A wide range of standard kernel functions can be used with support vector machines including:

Linear kernel $\text{kernel}(\mathbf{d}, \mathbf{q}) = \mathbf{d} \cdot \mathbf{q} + c$

where c is an optional constant

Polynomial kernel $\text{kernel}(\mathbf{d}, \mathbf{q}) = (\mathbf{d} \cdot \mathbf{q} + 1)^p$

where p is the degree of a polynomial function

Gaussian radial basis kernel

$\text{kernel}(\mathbf{d}, \mathbf{q}) = \exp(-\gamma \|\mathbf{d} - \mathbf{q}\|^2)$

where γ is a manually chosen tuning parameter

- 1 Interpreting Multivariable Linear Regression Models
- 2 Setting the Learning Rate Using Weight Decay
- 3 Handling Categorical Descriptive Features
- 4 Handling Categorical Target Features: Logistic Regression
- 5 Modeling Non-linear Relationships
- 6 Multinomial Logistic Regression
- 7 Support Vector Machines